Escuela Politécnica Superior

Pachangas App

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
Alejandro Bonafonte Compañ

Tutor/es:
Ester Martínez Martín

Junio 2018

Universitat d'Alacant
Universidad de Alicante

1. Justificación y Objetivos

El proyecto de PachangasApp nace con la idea de automatizar y simplificar la gestión de los partidos de fútbol 7 entre compañeros de trabajo. Antes del desarrollo de la app, el seguimiento de los partidos, así como la clasificación de los jugadores según sus resultados en los partidos, se estaba realizando mediante una hoja de cálculo online. Así pues, una aplicación móvil respaldada por un servidor que centralizase los datos podría aportar una mejora, tanto en gestión de los datos como en la visualización de los elementos por parte de los jugadores. Además ofrecería la oportunidad de ampliar las funcionalidades, como, por ejemplo, notificaciones a los participantes.

El proyecto que se va a desarrollar será crear una aplicación móvil que sirva para gestionar las ligas y partidos de fútbol entre amigos y poder visualizar las clasificaciones y partidos. Los usuarios registrados podrán crear ligas, temporadas y partidos, agregar a otros usuarios a las temporadas para que puedan apuntarse a los partidos y que los usuarios puedan ver los partidos, las clasificaciones (de puntos y goles) y los jugadores de las temporadas en las que están inscritos. Esta aplicación se comunicará con un servidor web para centralizar los datos.

Los objetivos a conseguir en el sistema de PachangasApp podemos clasificarlos en dos categorías: funcionales y técnicos. Por un lado, como objetivos funcionales que se quieren conseguir, es tener una aplicación móvil que tenga las características explicadas en el párrafo anterior y proporcionar un portal web alternativo a la aplicación Android para los usuarios que no tengan este sistema operativo. Además, el servidor web debe ser capaz de poder enviar notificaciones a la aplicación móvil y emails a los usuarios registrados para indicar nuevos eventos. Por otro lado, los objetivos técnicos son: crear un sistema de cliente/servidor (aplicación móvil en Android y servicio web en Apache/PHP) utilizando la infraestructura REST; tener el código, tanto de la aplicación móvil como del servidor, bien estructurado en capas y con un conjunto importante de tests para asegurar el buen funcionamiento del sistema; y que la aplicación móvil sea rápida y eficiente, tanto a la hora de procesar los datos como al solicitarlos al servidor.

2. Agradecimientos

Me gustaría agradecer a la Universidad de Alicante y a los profesores del Máster Universitario en Desarrollo de Software para Dispositivos Móviles por dotarme de los conocimientos necesarios para llevar a cabo este trabajo; a la tutora que me han ayudado en el desarrollo del mismo; a Víctor por su ayuda en el testeo final de la aplicación y finalmente a mi pareja Laura, por el tiempo robado durante todo el master.

3. Índices

3.1. Índice de contenidos

3.2. Índice de figuras

4. Introducción

Ante la necesidad de mejorar el sistema de partidos y clasificación que se llevaba entre los compañeros de trabajo (un excel online), se decide crear una aplicación móvil donde, de manera sencilla, se pueda crear ligas e invitar a amigos a participar.

La aplicación PachangasApp podrá registrar nuevos usuarios proporcionando por parte del usuario unos datos básicos (email, alias y password). Estos usuarios, al registrarse podrán crear ligas, estas ligas podrán tener temporadas y cada temporada tendrá un conjunto de partidos. El usuario podrá ponerle nombre a las ligas y temporadas. A las temporadas, además, se le podrá asociar jugadores (virtuales y reales).

Los jugadores virtuales son jugadores creados por el usuario y que están asociados a la temporada. Estos jugadores serían los que no tienen la aplicación móvil ni se quieran registrar en el sistema. Contabilizarán para las clasificaciones pero es, el administrador de la liga, el que debe añadirlos a los partidos y comunicarles personalmente cuando y donde se juegan. Los jugadores reales son otros usuarios de la aplicación que se han inscrito a la temporada. Estos jugadores, además de las mismas características que los jugadores virtuales, tienen la diferencia que reciben notificaciones de los partidos y que pueden apuntarse ellos mismos a los partidos por medio de la aplicación.

Los partidos se crearán con unos datos básicos (fecha y hora del día del partido, ubicación del campo donde se juega y si se ha jugado o no) y se podrá añadir jugadores de la temporada, tanto virtuales como reales. Además, el usuario podrá incorporar jugadores temporales para el partido en concreto. Estos jugadores que añada, no contabilizarán para las clasificaciones de la temporada.

Los usuarios, aparte de administrar las ligas que hayan creado, también podrán visualizar los datos de otras ligas creadas por otros jugadores en las que estén inscritos. El usuario que administra una temporada podrá invitar a otros jugadores, mediante email, y estos jugadores podrán aceptar o declinar dicha invitación por la aplicación o por email.

La aplicación se decide desarrollar bajo el sistema operativo Android debido al gran porcentaje de usuarios que lo utilizan, un 80% [4], y a las potentes funcionalidades que nos proporcionan tanto el sistema como la comunidad que hay detrás de este. Con el fin de integrar a aquellas personas no usuarias de Android, se decide crear un portal web donde también se pueda visualizar las ligas y resto de datos sin necesidad de tener la aplicación instalada.

5. Marco teórico

En la decisión de crear la aplicación se valoran diferentes cuestiones que afectan a la hora de llevar a cabo el sistema.

Para llegar al mayor número de usuarios, se decide desarrollar una aplicación móvil en Android ya que es el sistema operativo para móviles con mayor número de usuarios actualmente.

Como la aplicación móvil tiene que poder compartir datos (partidos, temporadas, etc.) entre los diferentes usuarios que la utilicen y además, se quiere tener un portal web donde consultar la misma información, se procede a hacer un sistema de cliente/servidor mediante REST para aprovechar la infraestructura HTTP[1] . Para centralizar la lógica de la aplicación y no tenga que estar duplicada en la aplicación móvil, portal web y otros futuros sistemas, es en el servidor donde se realizarán todos los cálculos y se persistirán los datos de la aplicación. Esto ayuda a que la aplicación cliente (sea cual sea el sistema) sea más ligera y solo se encargue de mostrar los datos, ayudando a un menor número de errores y duplicidad de código.

En el desarrollo de una aplicación móvil Android, es típico realizar un código, que aunque funcional, está muy acoplado, es poco reutilizable y testeable. Uno de los mayores problemas es crear todo el código de la aplicación en las actividades o fragmentos. Para solucionar este problema, se aplicará una arquitectura MVP (Modelo-Vista-Presentador) que permite tener el código desacoplado y con una sola responsabilidad[5]. Además, se implantarán otras buenas prácticas en la construcción de software, como la inyección de dependencias, que facilita la reutilización de código y testeo del mismo.

Igual que en la aplicación móvil, en la aplicación web también se realizará implementando una arquitectura MVC (Modelo-Vista-Controlador) que pueda ayudar a reutilizar la lógica del sistema (Modelo) tanto en la API como en el portal web. Aunque existen diferentes frameworks que ayudan a implementar estas características, uno de los más utilizados en la actualidad, tanto por su funcionalidad como por su eficiencia, es Laravel[3].

Para la persistencia de datos en la parte web, se decide utilizar el sistema de gestión de bases de datos MySQL[6], una base de datos con licencia pública con un gran número de usuarios y unas características que cumplen los requisitos del sistema.

6. Objetivos

El objetivo principal del proyecto es crear una aplicación móvil en Android con la que administrar de manera sencilla y rápida partidos de fútbol que se realizan entre compañeros y amigos de manera frecuente. Esta aplicación proporcionará al usuario la opción de crear ligas y partidos, inscribiendo a jugadores a los eventos para que se pueda llevar a cabo una clasificación según los partidos que se vayan jugando. La aplicación se encarga de calcular dicha clasificación. Además se podrán inscribir a esas ligas otros usuarios que podrán ver las clasificaciones, partidos jugados y recibirán notificaciones de próximos partidos que se creen.

Uno de los objetivos secundarios es poder implantar los aspectos aprendidos durante el Máster en una aplicación real. Como aplicar todo el temario no es viable (ni tampoco tenía sentido para la aplicación concreta) se buscan los siguientes:

- Conocimientos básicos de una aplicación Android (hilos de ejecución, navegación entre actividades, fragments, menús, notificaciones, etc.)
- Persistencia en aplicación móvil mediante *Shared Preferences*
- Implantar un servidor web con Laravel que proporcione tanto una página web como una API REST
- Consumir una API REST en Android
- Aplicar arquitectura MVP en Android
- Utilización de servicios proporcionados por Google: *Firebase Cloud Messaging* y *Maps*
- Publicar un *apk* en *Play Store*

Otros objetivos secundarios, más allá de lo aprendido en clase pero necesarios para el correcto funcionamiento del sistema, son:

- Instalación y configuración de un servidor web PHP con Mysql y con comunicación HTTPs.
- Utilización de librerías en Android que faciliten la inyección de dependencias: Dagger2[2] .
- Realización de testing automático tanto en la aplicación Android como en el API REST del servidor.

7. Metodología

Para que un proyecto se lleve a cabo de manera correcta, es necesaria una buena metodología de trabajo. Aunque los objetivos estaban claros al inicio del proyecto, según lo que se fuera aprendiendo durante el curso era posible que se añadieran algunas funcionalidades y se descartasen otras. Las metodologías ágiles han demostrado que dan mejores resultados que proyectos en cascada[7]. A pesar de ello se llevó a cabo un primer análisis de las características que se iban a implementar y de la arquitectura deseada.

Para este proyecto se ha seguido un desarrollo incremental del producto con lo que en las primeras iteraciones del producto ya se podía tener la aplicación en funcionamiento.

Durante las primeras fases del proyecto se ha llevado a cabo la creación y configuración de un servidor en la nube para poder centralizar los datos y que la aplicación móvil se pueda comunicar con él.

En estas primeras fases también se optó por la investigación de arquitecturas y diseños de programación que dieran más robustez y calidad al código que se iba a implementar. Aquí se pueden diferenciar dos partes:

- **Servidor**: Para el servidor se decidió implantar lo aprendido en el Máster en la asignatura de Programación Hipermedia para Dispositivos Móviles, es decir, implementar un servidor web en PHP utilizando el framework Laravel. Laravel nos facilita el desarrollo de aplicaciones web. Se basa en el patrón MVC y nos proporciona otras características que ayudan a la implementación como sistema ORM, gestión de bases de datos, etc. Además, gracias a utilizar el patrón MVC se ha podido implementar, tanto la página web como una API, reutilizando la lógica de la aplicación y además tener una mayor cobertura de tests. Para implementar la API que permite la comunicación entre la aplicación móvil y el servidor se ha decidido usar la arquitectura REST utilizando JSON como el formato del contenido dada su ligereza. Para mantener la calidad durante el desarrollo se han realizado test automáticos que cubren las funcionalidades que ofrece la API.
- **Aplicación móvil**: Se ha desarrollado una aplicación móvil en Android enfocada a teléfonos móviles. Se ha profundizado en investigar e implementar la arquitectura MVP para tener un código desacoplado y que se pueda probar fácilmente. Se ha intentado realizar continuas pruebas automáticas mientras se hacía el desarrollo para tener una importante cobertura del código.

Una vez establecida la arquitectura tanto de la aplicación móvil como del servidor se continuó con el desarrollo del sistema. Para ello se trabajó en paralelo en el servidor y en la aplicación móvil.

Mediante cortes verticales de todo el sistema, se han ido implantando las características que se consideraban oportunas y que podían dar más valor:

1. Se planteaba que característica implementar y de cómo consumirlo en la API.
2. Se desarrollaba la funcionalidad en el servidor.
3. Se realizaba el desarrollo en la aplicación móvil.

De esta manera siempre que se realiza una modificación en el sistema se aporta valor y se tiene una versión estable y funcional.

8. Cuerpo del trabajo

Como se ha comentado a lo largo de este documento, el sistema de PachangasApp se compone de dos módulos que se comunican entre sí basándose en la arquitectura cliente/servidor. Por un lado está la aplicación móvil que sirve de interfaz para el usuario y por otro lado el servidor web que centraliza los datos. La comunicación entre ambos módulos se realiza utilizando la arquitectura REST y como formato del contenido, JSON. El servidor se ha instalado bajo el protocolo https, por lo que la comunicación entre ambas partes está cifrada.

8.1. Aplicación Móvil

La aplicación móvil se ha realizado para el sistema operativo Android. La aplicación está compuesta de actividades y fragmentos para dotar de la funcionalidad deseada.

Estas actividades y fragmentos heredan cada una de una clase base que aglutina código que deben tener todas las clases.

Como se ha comentado anteriormente, todas las pantallas se basan en el patrón MVP, por lo que para cada Actividad o Fragment tenemos:

- **Layout**: Representación en xml del diseño de la pantalla. Definiendo la estructura de los elementos visuales en el layout tenemos una separación completa del diseño y organización de los componentes del código, pudiendo tener diferentes pantallas según el idioma del móvil o la inclinación.
- **Activity/FragmentView**: Código de la vista de la pantalla. Estas clases heredan de Activity o Fragment y tienen el código mínimo y necesario para tratar los eventos de la pantalla y configurar los objetos que se muestra dinámicamente: textos, si están visibles o no, habilitados, etc. Esta clase no tiene lógica, siempre está a la espera de las acciones del usuario o de la llamada del *Presenter*.
- **Presenter**: Controlador que se encarga de actualizar la vista y de llamar al *Modelo* cuando es necesario. Esta clase no contiene dependencias con el framework de Android, por lo que es fácilmente probable de manera independiente.
- **Modelo**: Debido a que la aplicación deja toda la lógica en manos del servidor, en PachangasApp tenemos una clase que se encarga de realizar la comunicación por REST al servidor. Toda la comunicación se realiza de manera asíncrona, por lo que las respuestas del servidor se informan al *Presenter* mediante callbacks.
- **Adapters**: Para el manejo de listas personalizadas en las pantallas de Android se deriva la responsabilidad del funcionamiento en los Adapters[9]. Éstos se encargan de capturar los eventos de las listas y mostrar la información. Para obtener la información necesaria, se apoyan en una dependencia que es el Presenter. En casi todos los elementos de lista se ha

utilizado el elemento RecyclerView[12], haciendo que sea necesario crear los adaptadores para su manejo.

Aunque finalmente las vistas y adapters llaman al presenter y éste al modelo, todas las relaciones se basan en interfaces, por lo que el acoplamiento es mínimo, permitiendo en cualquier momento cambiar un módulo/clase sin que afecte a las otras partes. Por ejemplo, para los test visuales, se ha procedido a inyectar en los presenter un *mock* del modelo, por lo que la aplicación funciona como queremos en cada caso sin realizar ninguna llamada al servidor. Esta inyección se ha realizado mediante el framework *Dagger2*.

Quitando las actividades iniciales (Splash, Login y Registro), la aplicación se basa en una sola actividad central encargada del menú de navegación (*Navigation Drawer*) y por cada opción del menú tenemos un fragment para cada pantalla.

A continuación procedemos a explicar cada pantalla.

8.1.1. Splash

Al iniciar la aplicación se muestra una pantalla de Splash mientras se realiza una comunicación con el servidor. De esta forma, al acceder a la aplicación ya hemos comprobado que la comunicación es correcta. Además se deja esta pantalla para, en un futuro, precargar datos que puedan acelerar el funcionamiento posterior de la aplicación.

8.1.2. Login

Si la comunicación es correcta se muestra la pantalla de Login.

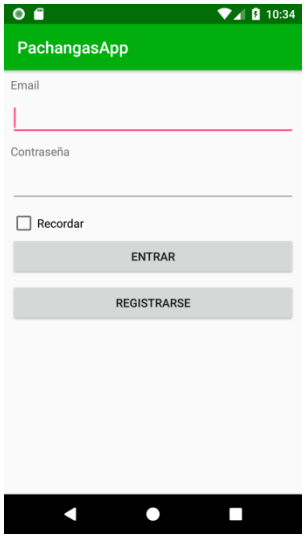

Figura 1. Login

En esta pantalla, Figura 1, si se está registrado previamente, se puede insertar el email y password con el que hemos realizado el registro. Además, marcando el checkbox *Recordar*, la aplicación

persiste en las preferencias de la aplicación los datos de *login* y, en futuros accesos a la aplicación, no hará falta volver a escribir el usuario y contraseña, entrando automáticamente a la pantalla *Home*.

Al pulsar en *ENTRAR* se enviarán los datos al servidor para comprobar si son correctos. En caso afirmativo se accederá a la pantalla de *Home*.

Si el usuario no se ha registrado previamente puede registrarse pulsando el botón *REGISTRARSE*.

8.1.3. Registro

Si el usuario no está registro puede acceder a la pantalla de registro, Figura 2, donde se le solicitarán unos datos básicos: email, un alias que se mostrará en la aplicación y una contraseña para acceder.

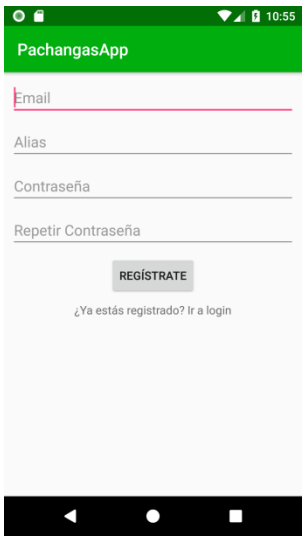

Figura 2. Registro

Si los datos introducidos son correctos (email válido, contraseña con tamaño mínimo, etc.) se muestra un mensaje que el registro se ha realizado correctamente y automáticamente se vuelve a la pantalla *Login*.

8.1.4. Home

La pantalla *Home* puede tener dos aspectos diferenciados. Si el usuario tiene ligas o si aún no ha creado ninguna. Al acceder por primera vez, en la pantalla se muestra una explicación de la aplicación y dos botones para ayudar al usuario a comenzar (Home sin ligas), Figura 3. Por otro lado, cuando el usuario ya tiene alguna liga, en la pantalla de Home se muestran cuatro categorías (Home con ligas), Figura 4:

- Próximos Partidos
- Últimos Partidos

- Clasificación
- Máximos goleadores

Estos cuatro apartados son un resumen de los datos que se obtienen en las diferentes pantallas que se explicarán en los siguientes apartados. En esta pantalla, se pide de manera asíncrona los datos para mostrar las categorías mencionadas. En *próximos partidos* y *últimos partidos* se muestran los partidos de todas las ligas/temporadas del jugador, mientras que en *clasificación* y *máximos goleadores* solo se muestran los datos de la liga/temporada que tenga elegida el usuario.

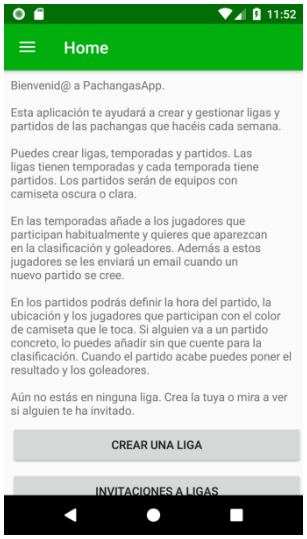

Figura 3. Home sin ligas

Figura 4. Home con ligas

8.1.5. Seleccionar Liga/Temporada

Como se ha comentado, para mostrar la clasificación y la tabla de goleadores, el usuario debe elegir una liga y temporada que será la que se usará para obtener los datos de estas dos pantallas.

Se obtendrá desde el servidor un listado de las temporadas (con el nombre de la liga) para que el usuario seleccione la que desee mostrar. Una vez seleccionada la temporada que se desee, se vuelve a la pantalla *HOME*. Figura 5

Figura 5. Seleccionar temporada

8.1.6. Partidos

En la pantalla *Partidos de la temporada*, Figura 6, se muestran los partidos de la temporada seleccionada. Por cada partido, se lista la fecha del partido, el resultado del mismo y un icono indicando de qué camiseta iba el usuario. Además, para los partidos sin jugar, aparece un elemento *switch* que el usuario puede pulsar para indicar si asistirá al partido o no, enviando la opción elegida al servidor inmediatamente.

Pulsando sobre un partido se puede ver información ampliada de este.

Figura 6. Partidos temporada

8.1.7. Ver Partido

En esta ventana se obtienen todos los datos del partido y se muestran para que un jugador tenga toda la información posible. A parte de la fecha y el resultado que también se mostraban en la pantalla de partidos, se muestra en que equipo ha jugado (o va a jugar) cada jugador y los goles que

ha marcado, Figura 7. Lo normal es que cuando se vayan apuntando los jugadores al partido no se les ponga en un equipo u otro, por lo que se muestra un listado de los jugadores que aún no tiene un equipo.

Finalmente, si se ha guardado la ubicación del partido, se muestra un mapa de *Google Maps* y un marcador indicando dónde se juega. Para ello, se ha utilizado la API de mapas de Google. Con esto, el usuario puede fácilmente utilizar la funcionalidad de Google para recibir las indicaciones para ir al partido.

En la siguiente imagen se puede ver la distribución de la pantalla.

Figura 7. Ver partido

8.1.8. Clasificación

Igual que los partidos de la temporada, la clasificación también se muestra sólo si has seleccionado una temporada. A diferencia de la pantalla *HOME* que se muestran los tres primeros clasificados, aquí se muestra la clasificación de todos los jugadores. Según como se oriente el dispositivo se muestran más o menos datos en la clasificación. En modo vertical, Figura 8, solo se muestra el alias del jugador, los puntos y los partidos ganados, empatados y perdidos. Por el contrario, si el usuario gira el móvil y lo pone en modo horizontal, Figura 9, se muestran todos los datos que se guardan actualmente, ampliando a los datos anteriores los goles, la racha de victorias y los porcentaje de victorias y de asistencia a los partidos.

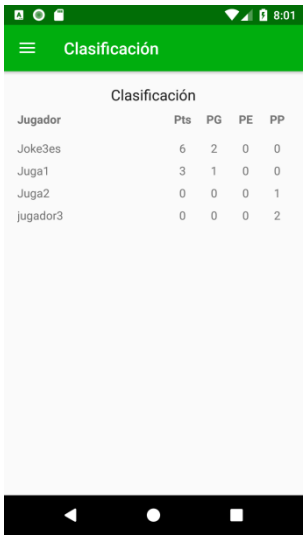

Figura 8. Clasificación vertical

Figura 9. Clasificación horizontal

Los datos obtenidos son los mismos desde el servidor, y lo único que varía en este MVC es el layout que se utiliza.

8.1.9. Goleadores

En la pantalla de Máximos Goleadores, Figura 10, se muestra un listado de todos los jugadores de la temporada que han marcado gol indicando el alias del jugador y los goles hasta el momento.

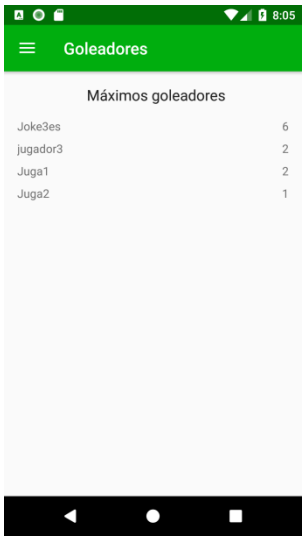

Figura 10. Goleadores

8.1.10. Jugadores

Finalmente, de las pantallas que son puramente informativas, tenemos el listado de jugadores, Figura 11. Simplemente se muestra un listado de todos los alias de los jugadores que participan en la temporada.

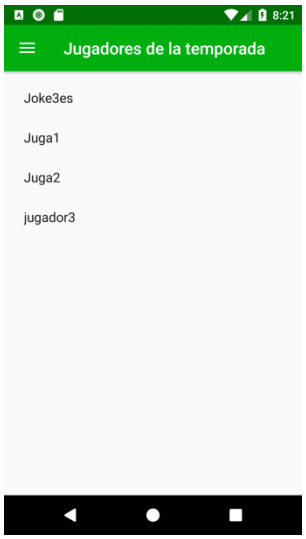

Figura 11. Jugadores

8.1.11. Entrar por invitación

Continuando con el orden descendente de las pantallas en el menú, se encuentra la pantalla donde se listan las invitaciones que un jugador recibe para inscribirse en ligas que han creado otros jugadores, Figura 12.

Se muestra el nombre de la liga y temporada junto a un *switch* para que el usuario indique si acepta la invitación. Si activa el componente inmediatamente se envía al servidor la decisión y el jugador empieza a participar en la temporada, pudiendo ver los partidos, clasificación y demás categorías.

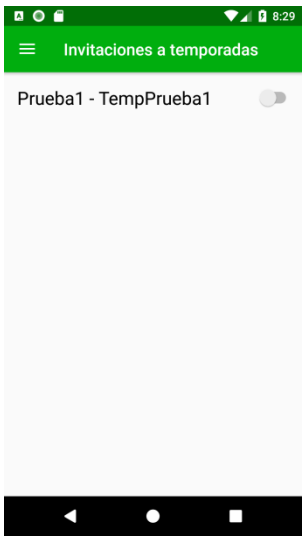

Figura 12. Invitaciones a liga

8.1.12. Administrar Ligas

En PachangasApp, cualquier jugador puede crear sus ligas ya sea para compartir con otros usuarios o para llevar los datos internamente. Para crear una liga, el usuario debe entrar en la pantalla Administrar Ligas y pulsar el botón flotante +.

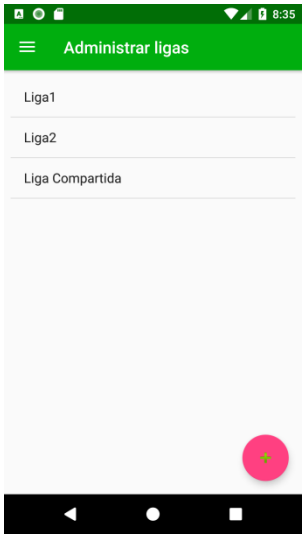

Figura 13. Administrar ligas

Si el usuario ya había creado ligas con anterioridad, se muestran en un listado. Como se ha comentado, pulsando el botón + o pulsando en alguna liga creada se accede a la pantalla donde el usuario puede crearla y modificarla.

8.1.13. Editar Liga

En esta ventana se puede crear o editar una liga. Actualmente la liga solo tiene el nombre editable. Además de eso se muestra el listado de temporadas que tiene la liga y se pueden añadir nuevas con el botón flotante +.

Esta ventana tiene un menú superior en el que se muestra el icono *check* para guardar la liga. Además, oculta se encuentra la opción de Borrar, que si la liga no contiene ninguna temporada se borra del servidor.

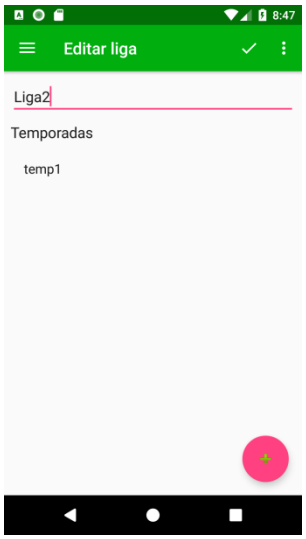

Figura 14. Editar liga

8.1.14. Editar Temporada

Ya sea creando una nueva temporada o editando una creada, se accede a esta ventana. En ésta, muy parecida a la anterior, se puede editar el nombre de la liga y se muestra un listado de los partidos de la temporada. Igual que en las ventanas anteriores, el botón flotante + permite crear nuevos partidos.

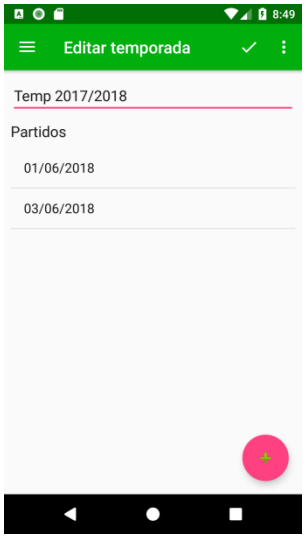

Figura 15. Editar temporada

En el menú de esta pantalla también está la opción de guardar, que envía los datos de la temporada al servidor y la opción de Borrar que elimina la temporada si no tiene partidos creados. Además se incluyen dos opciones más: *Jugadores*, que muestra los jugadores de la temporada y permite añadir nuevos; y *Calcular Clasificación*, que manda la acción al servidor y éste calcula automáticamente la clasificación de los partidos que están marcados como jugados.

8.1.15. Jugadores de temporada

Accediendo a la opción Jugadores desde el menú de *Editar Temporada*, se muestran los jugadores que actualmente están en la temporada, Figura 16. Éstos son los jugadores que se contabilizarán para la clasificación y los goleadores.

Estos jugadores pueden ser de dos tipos: jugadores que están usando la aplicación o jugadores virtuales, es decir, unos alias que ha creado el usuario que administra la liga.

Desde esta ventana se pueden crear nuevos jugadores virtuales, escribiendo el nombre y pulsando el botón +.

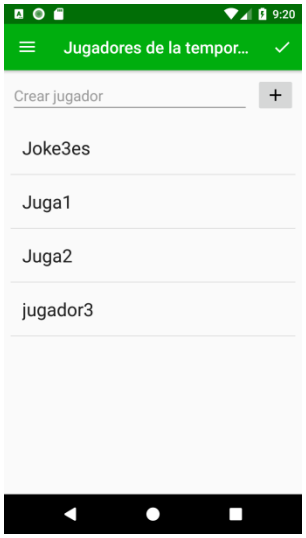

Figura 16. Jugadores temporada

Pulsando en cada uno de los jugadores se accede a la ventana de ver jugador, Figura 17, donde se puede ver más información del jugador.

8.1.16. Ver jugador

En esta ventana se muestran los datos del jugador: nombre, alias y correo electrónico.

Si el jugador es un jugador virtual como se ha explicado en el apartado anterior, se muestra un menú con la opción *Inscribir a temporada*.

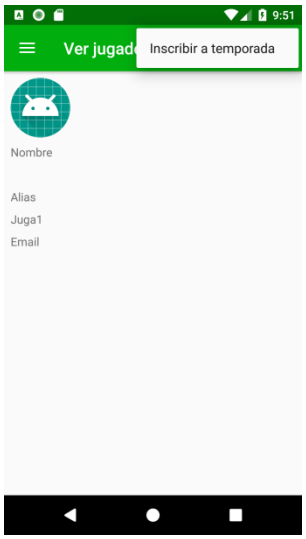

Figura 17. Ver jugador

8.1.17. Inscribir jugador a temporada

Si se desea que un jugador virtual se convierta en un jugador real mediante esta pantalla, Figura 18, es posible realizarlo. Se escribe el email del jugador y se selecciona la liga y temporada a la que se quiere inscribir de la lista disponible. Pulsando en *INSCRIBIR* se envía al servidor la petición y éste realiza las acciones oportunas. Aunque se profundizará más en secciones posteriores, si el email es de un jugador que ya existe en la aplicación, es decir, un jugador ya registrado, dicho jugador recibirá en su móvil una notificación.

Figura 18. Inscribir jugador a temporada

8.1.18. Editar partido

Esta pantalla, Figura 20, se utiliza para crear y editar los partidos que se vayan a jugar. Se puede establecer el día y hora del partido y la posición GPS donde se jugará. Para ello se ha utilizado el

servicio de Google: Places API Web Service. Este servicio proporciona una forma simple y clara para ubicar una posición en Maps, Figura 19.

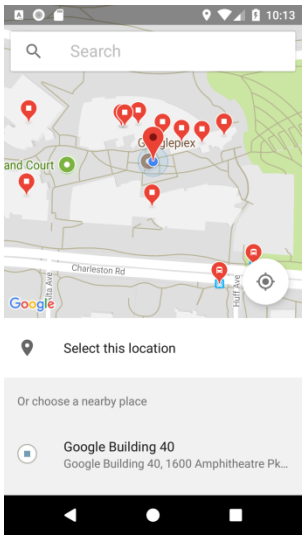

Figura 19. Places API Web Service

Además de estos datos, se puede indicar si el partido se ha jugado o no. Esto indicará si hay que tenerlo en cuenta para calcular la clasificación.

También se pueden establecer los goles de cada uno de los equipos (Oscuros y Claros).

Figura 20. Editar partido

Finalmente, se pueden añadir jugadores de manera manual al partido. Como se ha indicado anteriormente, los jugadores reales pueden marcar que asistirán al partido pero para los jugadores virtuales, es necesario que el administrador los añada manualmente. Además, el administrador debe seleccionar cada jugador en qué equipo juega (claro u oscuro) y los goles que ha metido.

Para añadir jugadores hay un *edit* de texto, Figura 21, en el que conforme se va escribiendo van saliendo los nombres de los jugadores que están en la temporada y al seleccionar se añaden a la lista de participantes

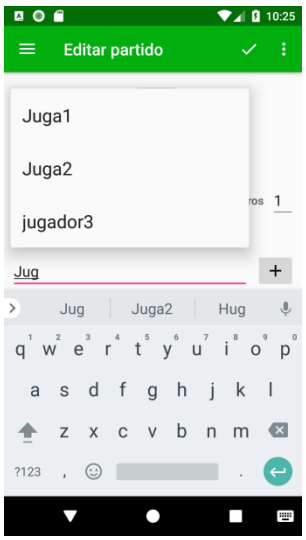

Figura 21. Editar partido II

Si al partido va a asistir un jugador que no asiste normalmente y no está en la temporada, se puede añadir al partido de forma puntual. Para ello, se escribirá el nombre del jugador y se pulsará en el botón +. Este jugador no contará para la clasificación ni saldrá en la tabla de goleadores.

8.1.19. Perfil

En la pantalla de Perfil salen los datos del usuario (email, alias y nombre) para que pueda editarlos si fuera necesario, Figura 22.

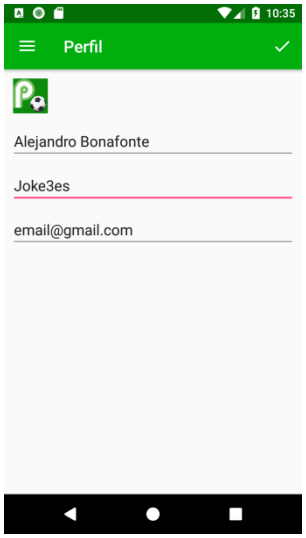

Figura 22. Perfil

8.1.20. Otros servicios

A parte de todas las ventanas indicadas con anterioridad, en la aplicación móvil también se encuentran otros servicios o acciones destacables.

- **Cerrar sesión**, para salir de la aplicación. Además, si se había marcado "Recordarme" al hacer login, también se borran los datos de inicio de sesión para que se vuelvan a solicitar
- **Servicios de notificación**, la aplicación puede recibir notificaciones push mediante el servicio de Google *Firebase Cloud Messaging*. Estas notificaciones se muestran para indicar si hay nuevos partidos o si el jugador ha sido invitado a una liga. Para poder enviar estas notificaciones, cuando el usuario accede a la aplicación (o mediante el servicio de *FirebaseIntanceIdService* que está recibe la notificación cuando se modifica el identificador) se envía al servidor el identificador del dispositivo que nos proporciona el framework de Firebase junto con el usuario actual para poder enviar notificaciones personalizadas.

8.2. Servidor

La aplicación móvil se ha creado para que sea un cliente con la menor lógica posible y es en el servidor donde se implementa la lógica de la aplicación y la persistencia de los datos. Esto nos aporta que se puedan desarrollar otros clientes para otros dispositivos reutilizando toda la API del servicio y tener una consistencia en todo el sistema.

Como se ha comentado, el servidor se ha desarrollado en un servidor Apache con PHP y utilizando el framework Laravel. La aplicación se ha instalado en un servidor virtual utilizando los servicios de Amazon [14].

Se ha instalado un certificado SSL autogenerado para encriptar toda la comunicación que haya entre cliente y servidor. Es por ello, que el acceso a este es mediante https.

Como sistema de gestor de base de datos se ha utilizado MySQL. Utilizando la funcionalidad de *migrations* que proporciona Laravel, se ha ido modificando la base de datos según las necesidades que se iban teniendo de manera ordenada y controlada. Finalmente, el esquema de la base de datos ha quedado de la siguiente manera, Figura 23:

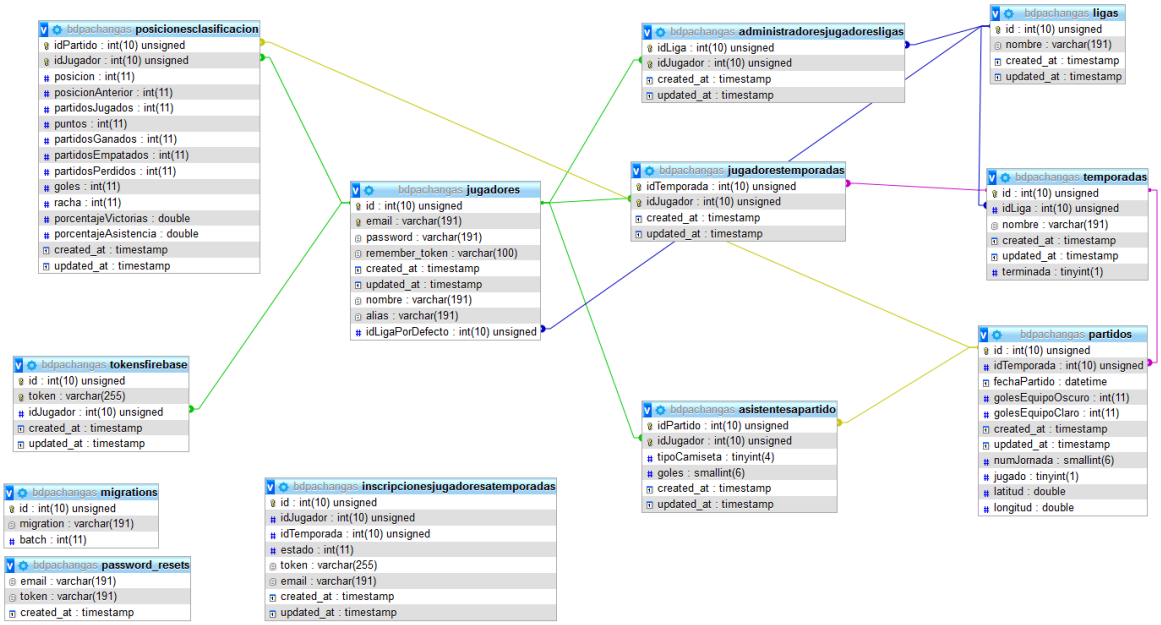

Figura 23. Esquema base de datos

Entrando en las funcionalidades del servidor, principalmente se persisten los datos proporcionados por la aplicación cliente en Android y se recuperan cuando se solicitan. Cada una de las pantallas de la aplicación móvil realizaba una llamada a un recurso del API, por lo que, aunque el trabajo de crear cada una de ellas ha sido significativo, no se va a detallar cada una de ellas. Se puede ver un listado completo en el anexo 11.1 API REST Servidor

Se van a destacar las funcionalidades más específicas o con mayor relevancia.

8.2.1. Middleware Autenticación

Se ha dotado a todo el API del servidor (exceptuando algunos recursos concretos) de un middelware de autenticación. Esta autenticación se basa en los usuarios registrados en la aplicación.

8.2.2. Guardar partido

Al guardar un partido, el servidor, aparte de persistir los datos recibidos del partido, envía una notificación a todos los jugadores de la temporada. Para ello utiliza el servicio Google Firebase Cloud Messaging. Utilizando la herramienta curl, envía una petición HTTP/POST al API de Firebase por cada uno de los tokens de los dispositivos almacenados y es Firebase quien envía la notificación al móvil. Como se ha comentado con anterioridad, cuando el usuario se loguea en la aplicación, se envía al servidor el token de Firebase con el usuario. De esta manera, se puede notificar personalmente a los jugadores cuándo hay nuevos partidos.

8.2.3. Calcular clasificación

La parte del sistema que tiene más lógica es el cálculo de la clasificación. Cuando se solicita calcular la clasificación de una temporada, el sistema recorre los partidos jugados y va calculando la clasificación de cada jugador dependiendo de los resultados y goles de los partidos disputados. Por cada partido ganado se le dota al jugador con 3 puntos y si ha empatado con 1 punto. Para ordenar la clasificación, se comprueba que jugador tiene más puntos, a igual número de puntos, se comparan los partidos ganados, luego empatados, goles, racha y finalmente partidos jugados. La clasificación se calcula y persiste partido a partido, por lo que da la posibilidad de obtener de forma inmediata el estado de la clasificación en cualquier partido.

8.2.4. Inscribir jugador a temporada

Igual que al guardar un partido, si el email que se desea inscribir en la temporada ya es de un jugador que está registrado, se envía al dispositivo del usuario una notificación, Figura 24, indicando que le han invitado a inscribirse en una temporada.

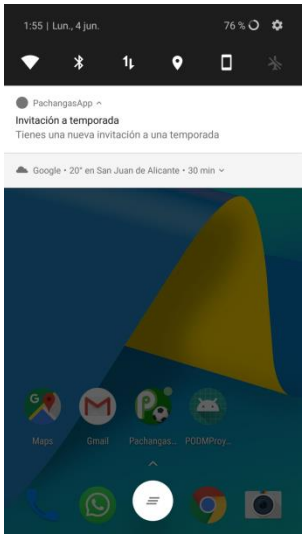

Figura 24. Notificación

Además de la notificación push, se le envía un email con un enlace donde puede pulsar para inscribirse automáticamente, Figura 25.

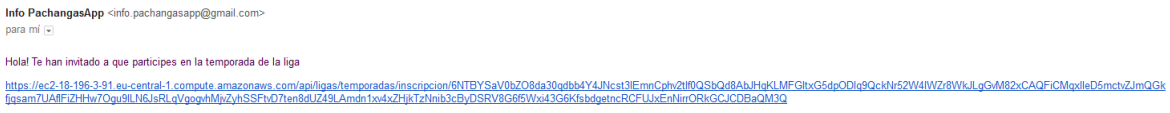

Figura 25. Email inscripción a temporada

8.3. Aplicación web

Aunque de manera muy simple, se ha dotado al sistema de un interfaz web donde poder consultar las ligas y partidos. Para ello, se ha continuado utilizando el framework de Laravel, pero aplicando una interfaz web en lugar de un API REST.

Accediendo a la url del servidor se muestra un formulario de login, Figura 26.

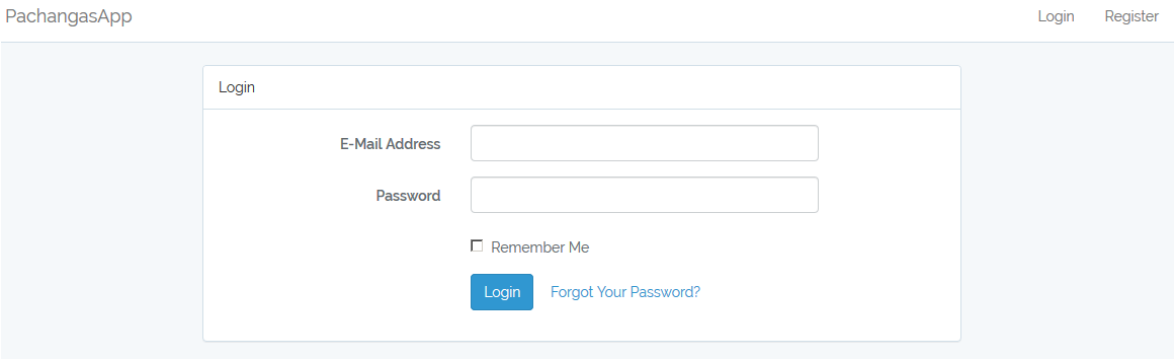

Figura 26. Login aplicacion web

Accediendo con el usuario y password con el que el usuario se haya registrado en la aplicación móvil, se puede ver todas las ligas, temporadas y partidos del usuario, Figura 27

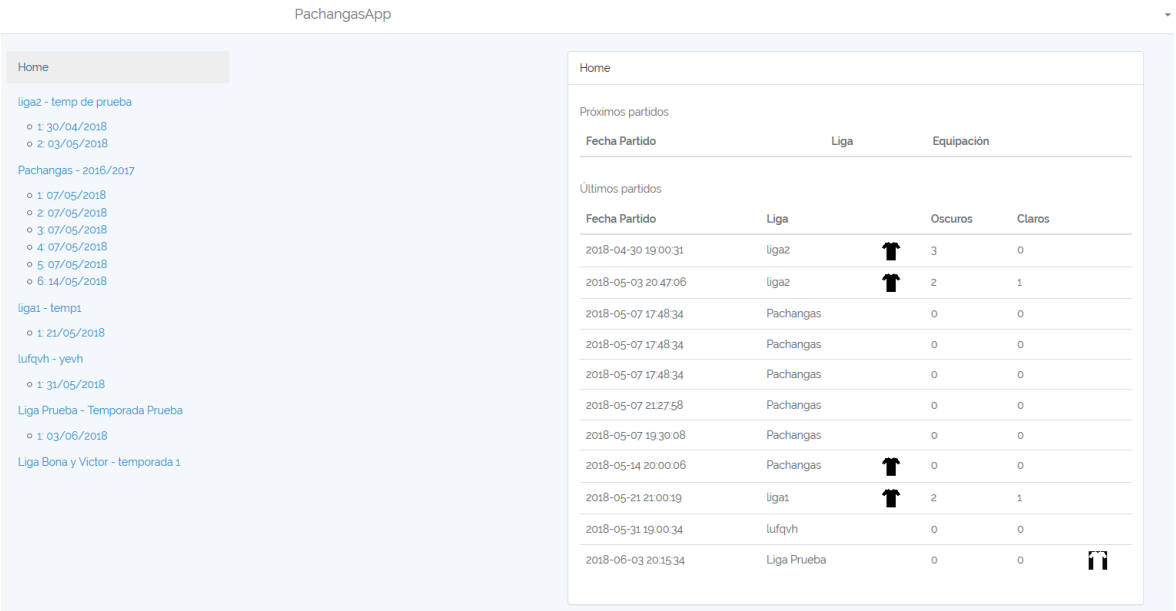

Figura 27. Aplicación web

Además, se ha dejado de manera pública para poder acceder a las ligas creadas. Actualmente se ha dejado abierto completamente, pero se pretende que el administrador establezca algún tipo de contraseña para que no pueda verlo cualquier persona. Esta funcionalidad está pensada para personas que sin registrarse, representen uno de los jugadores virtuales creados en la aplicación.

9. Conclusiones

El proyecto se ha llevado a cabo como se deseaba y se ha realizado la mayoría de las funcionalidades que estaban como objetivo.

Se ha creado una aplicación móvil que permite que un usuario pueda crear ligas, temporadas y partidos de fútbol y llevar una clasificación automática de los jugadores que participan según los resultados. Utilizando las notificaciones de *Google Firebase Cloud Message*, se informa a otros usuarios de nuevos partidos que se han creado.

Al crear un servidor para centralizar los datos y que la aplicación se comunique con éste, se permite compartir los datos entre las diferentes aplicaciones. Para la comunicación entre ambas plataformas se ha utilizado las diferentes llamadas a la API REST que proporciona el servidor.

En la aplicación móvil se ha utilizado las SharedPreferences de Android para almacenar datos concretos de la aplicación, en cambio, para la persistencia general del sistema se ha utilizado el servidor mediante MySql.

El servidor, aparte de proporcionar una centralización de los datos y la lógica del sistema, al ser un servidor PHP, también proporciona un portal web para mostrar la información de las ligas, temporadas y partidos, así como las clasificaciones.

La aplicación se ha publicado en *Google Play Store* en un segmento cerrado para que terceras personas pudieran descargase la aplicación y realizar pruebas alfa durante el desarrollo, ayudando a detectar errores y mejorar la usabilidad.

A pesar que se han conseguido las metas que impuestas al inicio del proyecto, quedan muchos puntos por abarcar y mejorar de la aplicación. Por un lado, mejorar el diseño y la usabilidad de la aplicación, haciendo que sea más intuitiva y sean más sencillas ciertas tareas como crear partidos o la invitación e inscripción a ligas de otros usuarios. Por otro lado, se debe aplicar un mayor grado de seguridad en el sistema, haciendo hincapié en la parte servidor. Finalmente, se debe mejorar la eficiencia de la comunicación entre cliente y servidor, para reducir los datos transmitidos entre ambos módulos.

10. Bibliografía

[1] Lozano Ortega, M., & Gallego Sánchez, A. (2017). *Desarrollo de aplicaciones Android con Java*. Madrid: Ra-Ma.

[2] https://github.com/google/dagger

[3] https://laravel.com/

[4] https://www.cnet.com/es/noticias/android-market-share-abril-junio-android-vs-ios-mercado-2016

[5] https://github.com/MindorksOpenSource/android-mvp-architecture

[6] https://www.mysql.com

[7] https://clearcode.cc/blog/agile-vs-waterfall-method/

[8] https://developer.android.com/index.html

[9] https://medium.com/@jsuch2362/adapter-what-role-is-it-data-view-13c713cdae0b

[10] https://android.jlelse.eu/android-handling-checkbox-state-in-recycler-views-71b03f237022

[11] https://es.stackoverflow.com/

[12] https://android.jlelse.eu/recyclerview-in-mvp-passive-views-approach-8dd74633158

[13] https://www.flaticon.com

[14] https://aws.amazon.com/es/

11. Anexo

11.1. API REST Servidor

Method	URI	Middleware
GET|HEAD	/	web,auth
GET|HEAD	api	api
PUT	api/firebase	api,log
PUT	api/firebaseAuth	api,log,auth.basic.once
POST	api/jugadores	api,log
GET|HEAD	api/jugadores/{email}	api,log,auth.basic.once
PUT|PATCH	api/jugadores/{jugadore}	api,log,auth.basic.once
POST	api/ligas	api,log,auth.basic.once
GET|HEAD	api/ligas/administradas	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas/administradas	api,log,auth.basic.once
PUT	api/ligas/temporadas/inscripcion	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas/inscripcion/{token}	api,log
GET|HEAD	api/ligas/temporadas/invitaciones	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas/partidos/{partido}/jugadores	api,log,auth.basic.once
PUT	api/ligas/temporadas/{idTemporada}/clasificacion	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas/{temporada}/jugadores	api,log,auth.basic.once
PUT	api/ligas/temporadas/{temporada}/jugadores	api,log,auth.basic.once
POST	api/ligas/temporadas/{temporada}/partidos	api,log,auth.basic.once
GET|HEAD	api/ligas/temporadas/{temporada}/partidos/{desc?}	api,log,auth.basic.once
PUT	api/ligas/temporadas/{temporada}/partidos/{idPartido}	api,log,auth.basic.once
DELETE	api/ligas/{idLiga}	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/administradores	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/clasificacion	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/goleadores	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/temporadas	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/temporadas/{idTemporada}/clasificacion	api,log,auth.basic.once
GET|HEAD	api/ligas/{idLiga}/temporadas/{idTemporada}/goleadores	api,log,auth.basic.once
PUT|PATCH	api/ligas/{liga}	api,log,auth.basic.once
POST	api/ligas/{liga}/temporadas	api,log,auth.basic.once
PUT	api/ligas/{liga}/temporadas/{temporada}	api,log,auth.basic.once
GET|HEAD	api/partidos/proximosPartidos/jugador/{idJugador}	api,log,auth.basic.once
GET|HEAD	api/partidos/temporada/{idTemporada}	api,log,auth.basic.once
GET|HEAD	api/partidos/ultimosPartidos/jugador/{idJugador}	api,log,auth.basic.once
DELETE	api/partidos/{idPartido}	api,log,auth.basic.once
PUT	api/partidos/{idPartido}/camisetaJugador	api,log,auth.basic.once
PUT	api/partidos/{idPartido}/jugadorAsiste	api,log,auth.basic.once
DELETE	api/temporadas/{idTemporada}	api,log,auth.basic.once
GET|HEAD	home	web,auth
GET|HEAD	home/show/{id}	web,auth

GET|HEAD	ligas/{idLiga}	web
GET|HEAD	ligas/{idLiga}/temporadas/{idTemporada}	web
GET|HEAD	ligas/{idLiga}/temporadas/{idTemporada}/partidos/{idPartido}	web
POST	login	web,guest
GET|HEAD	login	web,guest
POST	logout	web
POST	password/email	web,guest
GET|HEAD	password/reset	web,guest
POST	password/reset	web,guest
GET|HEAD	password/reset/{token}	web,guest
GET|HEAD	publico	web,auth
GET|HEAD	register	web,guest
POST	register	web,guest